Beat Tracking

Eva Fineberg

March 7, 2021

Abstract

This assignment explores a beat tracking algorithm first proposed by Dan Ellis in 2007, and draws conclusions about it's performance. The algorithm is written to be tested against a data set consisting of a large variety of ballroom style music encompassing 10 different sub-genres. At a high level, the implementation accepts an audio file from this data set as input, and returns a series of tactus as the output. The tactus outputs were then evaluated for correctness and relevance against the known annotations data set [4]. In doing so, the findings conclude that the beat tracking algorithm performs best when estimating the *ChaChaCha* ballroom style music, and worst with *Samba.* The findings also show that the worst continuous beat estimation evaluated was for the *Waltz* style music files.

Context

Beat tracking, though subconsciously intuitive in human perception is a non-trivial task for computers. However, today, the computational equivalent of such a task is used across disciplines and implemented in many workspaces such as DAWs and other creative digital tools. *Beat tracking* "has the aim of recovering a sequence of beat onset times from a musical input consistent with human foot taps."[1] In order to achieve this aim, the times (or instances) of the *tactus* (or *primary pulse*) can be estimated using information derived from rhythmic properties of the music.

Rhythm is comprised of five key components: *Pulse*, *Metre*, *Tempo*, *Timing*, and *Grouping*, all of which are essential in understanding methods for beat tracking.[6] This assignment focuses on Dan Ellis'S *Beat Tracking by Dynamic Programming* which requires the use of data describing *tempo*, and *pulse*. *Tempo* is defined as "the rate of the primary pulse" and *pulse*, defined as "regularly spaced sequence of accents".[6] Ellis's approach relates these key characteristics with strength onset detection in order to extrapolate a beat sequence for a given music file.[2]

Implementation

In order to successfully estimate the beat in a given ballroom style music clip, this approach aims to re-implement Dan Ellis'S *Beat Tracking by Dynamic Programming* algorithm.[2] The goal of this algorithm is to find an optimal sequence of beats in an efficient manner. Dynamic Programming (DP) algorithms are optimisation algorithms, using the state of smaller sub-problems in order to efficiently reconstruct a solution for the original problem. In the context of beat-tracking, this algorithm considers subsections of a computed novelty curve and finds

the maximum value for a given prefix interval. These values are stored, and later used in look-up to estimate a beat sequence during back tracking.

In addition to the core beat tracking algorithm, multiple pre-processing steps were required in order to feed this algorithm data in an efficient shape. The procedural steps, which are outlined in detail below, are as follows: (1) Compute an onset detection function, (2) Enhance the onset detection function by compression, filtering and normalisation (3) Estimate the tempo of the given input audio file and finally (4) Using the values as strength from the detection function, estimate a beat sequence in the time domain.

Onset Detection

More specifically, this approach begins with an onset detection function used to describe an onset strength envelope. This will later be used as the core indication as to whether or not a tactus is present. The bulk of this work was done using librosa.[5] First, the spectral flux of the input signal was computed. This was done by evaluating the SFTF magnitude with a window length of 1024 samples and a $hope\_size = \frac{window\_length}{2}$. The sampling rate of the music was computed when the file input was first loaded.

Converting to a decibel scale, logarithmic compression was applied using the following formula:

$$X_{compressed} = log(1 + \gamma \cdot |stft|)$$

s.t. $\gamma = 100$. Finally, taking the sum of the first order difference, and saving only the positive difference values results in zeroing out drops in energy.[6]

Peak Enhancement

Following *Müller's* notes on *Ellis's* approach, we can then further improve improvements are made to the input signal. [6] [2] In order to further enhance the peaks in the resulting novelty (or onset) function, a local average window with a size M s.t. $M = 10$ was applied. The result of which was subtracted from the original onset function and subsequently normalised. This ultimately results in a more prominent onset strength envelope used to identify possible beat locations in relation to a given estimation. These pre-processing steps are done in order "to balance the perceptual importance of each frequency band"[6] and more closely approach how we as humans perceive the beat.

Dynamic Programming Approach

At the core of this approach, is the assumption that the global tempo of the input audio is already known. In order to satisfy this assumption the *librosa.beat.tempo* function was used.[5] The goal then is to generate a sequence of beats which align optimally with magnitude strengths in onsets as well as the estimated global rhythmic tempo of the audio input. Mathematically, can be expressed as the objective function:

$$C(t_i) = \sum_{i=1}^{N} O(t_i) + \alpha \sum_{2}^{N} F(t_i - t_{i-1}, \tau_p)$$

s.t. $F(\Delta t, \tau_p)$ represents the penalty (or confidence based on a loss function) of the estimate. More specifically:

$$F(\Delta t, \tau_p) = -\left(log \frac{\Delta t}{\tau} \right)^2$$

This penalty function measures how similar (or different) two consecutive beats are in comparison to the estimated beat sequence. The best score at a given time t, is the onset strength at that time summed with the maximum previous beat score. This backwards look-up is where the power of dynamic programming is apparent. By observing these two functions, it is apparent that the "best" scoring estimation exists when the penalty calculation is equal to zero, which would imply an identical onset strength and a perfectly spaced pulse metrical structure.

Results

This algorithm was run over the entirety of the ballroom data set [3]. The performance results of which were evaluated using the *mir_eval* library "which provides a transparent, standardised, and straightforward way to evaluate Music Information Retrieval systems".[7]

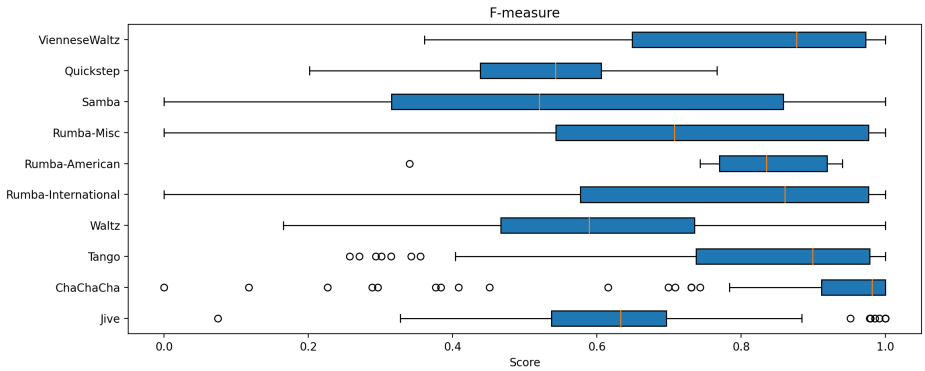

Figure 1. Box plot representation of the F-measure across sub-genres.
The f-measure estimates correctness with respect to the expected reference beat.

As shown in Figure 1, the implementation of this algorithm, with respect to "correctness", has performed slightly above average overall. The average F-measure across sub-genres is 0.69. The best beat sequence estimation was produced analysing the sub-genre *ChaChaCha* and the worst estimation was found on the *Samba* sub-genre.

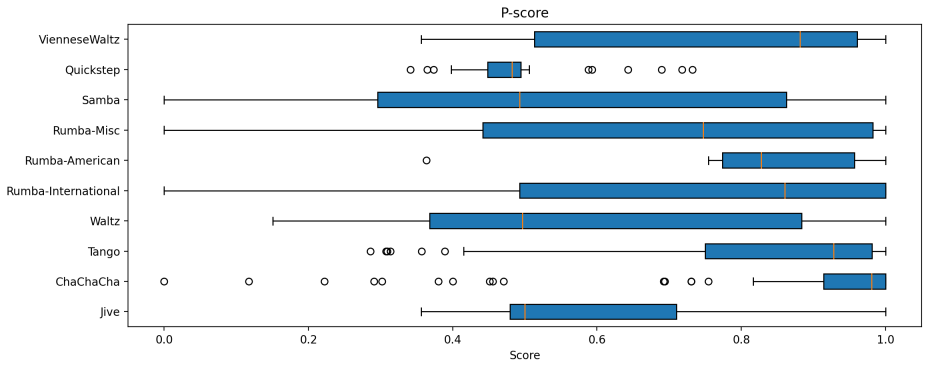

Figure 2. Box plot representation of the P-score across sub-genres.
The p-score computes the cross-correlation of the estimated and reference beat sequences represented as impulse trains.

The distribution of box plots on Figure 2 shows a P-score which exhibits similar behaviour to the F-measure with *ChaChaCha* outperforming all other sub-genres and *Samba* at the bottom.

This score evaluates the displacement of the estimated beat sequence relative to the reference sequence. The average P-score across sub-genres is 0.68.

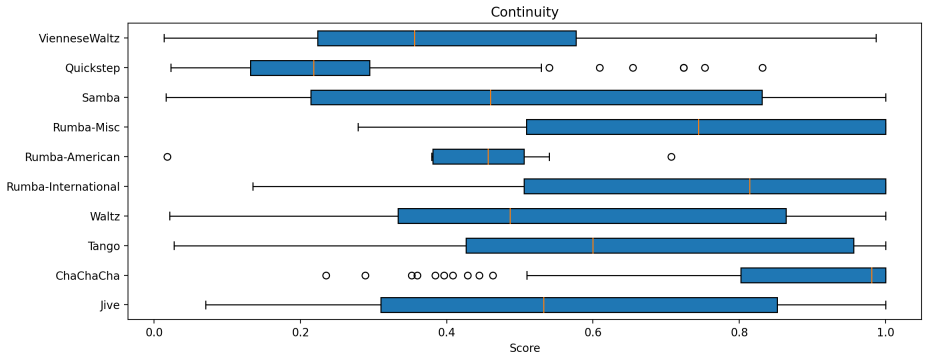

Figure 3. Box plot reprsentation of the Continuity score.
The continutiy score computes the proportion of the beat sequence which is continuously correct

Finally, we observe the continuity score of our propsed beat sequence which has the lowest cross-genre average score of 0.579. Similarly to our previous findings, the *ChaChaCha* sub-genre was estimated most accurately, but unlike the previous results, the *Viennese Waltz* sub-genre was estimated most poorly with regards to continuity.

These results are intuitively sensible, considering *ChaChaCha* style music has stronger, more consistent tactus in it's metrical structure and *Samba* often exhibits softer more synchopated patterns which would be more difficult for this algorithm to identify. Additionally, Waltz music, while known to usually keep a steady $\frac{3}{4}$ tempo, does not do so with emphatic percussion. Given the limitations of this algorithm, it makes sense this sub-genre has the worst continuity score.

Assumptions and Limitations

While this algorithm is efficient, and works impressively well for such a simple approach, there are some assumptions which provide limitations. The first is that this implementation assumes that the tempo is constant throughout an entire piece of music. This makes the model more rigid and less likely to understand the tempo of a more dynamic piece of music. Additionally, Ellis assumes that a large magnitude change in the onset strength envelope indicates there is a beat corresponding to a periodicity point in the tempo. However, with some genres tactus are not as strong, and so it is harder to recognise them as being part of the beat sequence.

References

[1] M. E. P. Davies and M. D. Plumbley. "Context-Dependent Beat Tracking of Musical Audio." In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.3 (2007), pp. 1009–1020. DOI: 10.1109/TASL.2006.885257.

[2] Daniel P. W. Ellis. "Beat Tracking by Dynamic Programming." In: *Journal of New Music Research* 36.1 (2007), pp. 51–60. DOI: 10.1080/09298210701653344. eprint: https://doi.org/10.1080/09298210701653344. URL: https://doi.org/10.1080/09298210701653344.

[3] Fabien Gouyon et al. "An experimental comparison of audio tempo induction algorithms," Trans." In: *on Speech and Audio Proc*, p. 2006.

[4] Florian Krebs. *Ballroom Annotations*. https://github.com/CPJKU/BallroomAnnotations. 2016.

[5] Brian McFee et al. *librosa/librosa: 0.8.0*. Version 0.8.0. July 2020. DOI: 10.5281/zenodo.3955228. URL: https://doi.org/10.5281/zenodo.3955228.

[6] M. Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing, 2015. ISBN: 9783319219455. URL: https://books.google.co.uk/books?id=HCI%5C_CgAAQBAJ.

[7] Colin Raffel et al. "$\text{mir}_e val$: $a transparent implementation of common MIR metrics$." In: *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. 2014.